Internal Reuse Breakout Session

Visualization Requirements Workshop
Bethesda, MD ~ June 2-3, 2003

Lead: Steve Parker
Scribe: Jim (a.k.a. Jeeembo) Kohl

Participants: Mike Papka, Mark Duchaineau

Basic Plan:

- Enumerate interfaces that viz tools need
  - Data models, graphics, color maps, transfer functions…
- List those in existence
- List reasons they are deficient

Distinguishing Our Breakout vs. "External" Breakout?

- Fine line between internal modules and external tool protocols…

Graphics

- OpenGL ~ efficient, because it doesn't deal with parallelism…
- Chromium
- DirextX/3D
- Volumizer
- Volumetric extensions to OpenGL
- Cg / Arb Shader…

Scene Graph ~ higher-level object management

- OpenRM
- OpenScenegraph
- OpenInventor
- Performer
- Java3D

GUI Toolkits

- Gtk
- Motif
- Glui
- Wxwindows
- Fltk
- Qt ~ warning: never use this…!

Viz Environments

- AVS
- OpenDX
- SCIRun
- VTK / ParaView
- Ensight

Data Models

- FM / FEL
- Silo
- GMV
- DMF (name not to be spoken)
- TSTT interface

I/O

- HDF5
- NetCDF

Data Management / Migration

- Htar
- Xftp
- SimTracker

Viz and Data Management often considered independently…

- This is a mistake. (SciDAC disconnect ~ SDM vs. Collaboratories)
- Viz often drives data management…!

Do we really want an "Über Data Format"?

- Perhaps not…
- Allow many distinct data formats with generic accessor interfaces…?
    - Callback-type interface?

Common Internal Viz Interfaces ~ "Extractions"…

- Streamline integration (as in integrals)
- Color maps / transfer functions
    - Color map representation ~ splines, bins, lines, gaussians…
- Isosurfacing
- Contouring
- Material Boundaries

- Particle to Field
- Slicing Planes / Dimension Reduction
- Resampling, Filtering
- Windowed Averaging / Smoothing
- Histograms, Contour Spectrum
- Statistical Analysis
- Topology Analysis
- Denoising
- Resource Management ~ memory, data caching…
- Feature Extraction / Tracking
- Transformations
    - Geometric
    - Wavelet
    - Fourier
- Hierarchy Building
- Chunking / Tiling, and Other Reordering
- Compositing
- Shader Languages
- Level of Detail Management
- Sorting (Geometric)
- Scripting
- Line Graphs, Plots… ☺
- Data Calculator
    - In support of a front-end user query language…
- Subsetting

These Areas Define the Universe…

→ What are the REAL problems…?

IF all these modules existed, and were interoperable, would this make the science case any better…?

Scientists want their customized, application-specific viz, with "5 buttons"…

- E.g. "heat transfer in automobile tires" button…
- Framework should be flexible to support a spectrum of people:
    - High-Level Application Scientist / Viz Support Person
    - Viz Developer
    - Bleeding Edge Viz Researcher

What are the benefits of interoperability?

- Faster time to delivery ~ rapid prototypes
- Consistent "look and feel" across independent development
    - A la Windows… you can guess what to do / how it works…

Are we covering all of the DOE science needs…?

- We mostly do big data simulations.
- Bio and info viz…?
- Particles as well as fields…?
- Representation Crossover: expand the horizons of application scientists…
    - Provide simple solutions that the scientists want, but try to enable evolution towards more sophisticated representations…!

The "Whole" Visualization Process Includes Data Analysis, Filtering, Processing…

- Starts from simulation output disk file, takes it all the way to final display…